KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems

Wiem Tounsi, Nora Cuppens-Boulahia, Joaquin Garcia Alfaro, Yannick Chevalier, Frédéric Cuppens

http://dx.doi.org/10.1016/j.jnca.2013.06.002

KEDGEN2: A key establishment and derivation protocol for EPC Gen2 RFID systems

Wiem Tounsi[1], Nora Cuppens-Boulahia[1], Joaquin Garcia-Alfaro[1,2], Yannick Chevalier[3], Frédéric Cuppens[1]

[1]*Institut Mines-Telecom, TELECOM Bretagne, CNRS Lab-STICC UMR 6285, France*
[2]*Institut Mines-Telecom, TELECOM SudParis, CNRS Samovar UMR 5157, France*
[3]*IRIT, Université de Toulouse, France*
{FirstName.SurName}@telecom-bretagne.eu,
joaquin.garcia_alfaro@telecom-sudparis.eu,
y.chevali@irit.fr

Abstract

The EPC Class-1 Generation-2 (Gen2 for short) is a Radio Frequency IDentification (RFID) technology that is gaining a prominent place in several domains. However, the Gen2 standard lacks of verifiable security functionalities. Eavesdropping attacks can, for instance, affect the security of applications based on the Gen2 technology. To address this problem, RFID tags must be equipped with a robust mechanism to authenticate readers before authorising them to access their data. In this paper, we propose a key establishment and derivation protocol, which is applied at both identification phase and those remainder operations requiring security. Our solution is based on a pseudorandom number generator that uses a low computational workload, while ensuring long term secure communication to protect the secrecy of the exchanged data. Mutual authentication of the tag and the sensor and strong notions of secrecy such as forward and backward secrecy are analysed, and we prove formally that after being amended, our protocol is secure with respect to these properties.

Keywords: Radio Frequency IDentification, Electronic Product Code, Cryptographic Protocol Verification, Model Checking.

1. Introduction

The Radio Frequency IDentification (RFID) technology is one of the most promising advances in current pervasive infrastructures [1]. It allows contactless identification of tagged objects and people, and has gained a prominent place on the retail industry and transportation systems [2, 3]. Today, it continues to gain places in even more sensitive contexts, like hospitals and inpatient care systems [4, 5]. The EPC Gen2 standard, short-hand for the Electronic Product Code (EPC) Class-1 Generation-2 [6], is a proper example of passive RFID technology. Gen2 scenarios rely on passive tags designed with very basic capabilities. Gen2 tags derive their transmission and computational power from the signal of an interrogating reader. Once a scan is done by the reader, the tag responds with a unique ID, called EPC, that is transferred to a back-end infrastructure for further processing. EPC serves as an identifier for the physical objects, locations, assets carrying the tag which, moreover, can be identified and tracked based on completely distributed architectures [7].

The security model of the Gen2 technology presents important drawbacks in sensitive contexts like healthcare systems [8]. Indeed, the security model of the Gen2 specification only considers limited protection of some special operations that require reader authentication, such as memory *access* and deactivation operations. These operations require the communication of a password prior the tag execution. This password must be sent via the insecure reader-to-tag channel. Since this channel is more likely to suffer from eavesdropping attacks than the tag-to-reader channel, the Gen2 specification proposes to protect the exchange as follows (Steps 3–5):

```
1.  READER  ⟶  TAG     :  RequestID
2.  TAG     ⟶  READER  :  TagID
3.  READER  ⟶  TAG     :  Key-request
4.  TAG     ⟶  READER  :  Key
5.  READER  ⟶  TAG     :  Password ⊕ Key
```

Steps 1 and 2 show a simplified *inventory* operation that we name identification stage, where reader asks the identity of the tag for further treatments. In Step 3, the reader informs that it is waiting for a key necessary to protect sensitive data in the following exchange (cf. Step 5). This exchange will eventually contain the required password to grant the execution of an *access* operation. The key is generated by the tag as a random bit string, and transmitted in Step 4 in plaintext to the reader. This is done via the tag-to-reader channel which, in principle, is expected to have an eavesdropping range much lower that the reader-to-tag channel. This exchange supposes that an adversary eavesdropping the reader-to-tag channel cannot capture the sensitive data (either the password or the contents of the password-protected operation). It is straightforward that an adversary capable of eavesdropping the tag-to-reader channel using special hardware devices (e.g., readers with high sensitive receivers and multiple antennas), or predicting the output of the

random bit generator of the tag (e.g., based on a flawed EPC Gen2 pseudorandom generator [9, 10]), can obtain the key and simply recover the access password by applying an Exclusive-OR operation. Obtaining the access password permits to the malicious adversary to access and modify the memory of the tag. When the memory of the tag includes additional information about objects which can be linked to people holding them (e.g., information related to a particular medicine which may link to a particular disease), information disclosure can be an important threat. Such a threat can be handled by modifying the security model of the Gen2 specification and providing a more elaborated way of managing cryptographic keys in Gen2 systems.

A second issue to address is the lack of formally verified security enhancements for Gen2 tags. While the community of cryptographic algorithms designers relies on mathematical proofs of the model they propose, the designers of RFID protocols and security architectures using lightweight cryptographic algorithms lack this sort of strong foundations. Thus, a major need of proved and validated security RFID protocols appears. These proofs may be done following a mathematical or logical calculation or using some tools developed for the purpose. Unfortunately, this was not systematic in most work dealing with RFID security where the correctness or security goals achievement were demonstrated *intuitively*. Recently many approaches focus on solutions to enhance the security of the EPC Gen2 technology, e.g., proposing physical solutions including new cryptographic primitives on-board of the tags or straightforward protocols that remain to be adapted to the Gen2 constraints. A great number of solutions have been reported as insecure. Recent cases of authentication techniques were reported vulnerable few time after their publication by [11, 12, 13]. These cases show the lack of formal verification of new security techniques for EPC Gen2, which we deem necessary [8, 14].

Motivated by these limitations (i.e., flawed Gen2 security model and lack of formally verified security enhancements), we propose in this paper a communication protocol designed to share and refresh security keys between readers and tags in Gen2 RFID environments. These keys encrypt security parameters and personal data when accessing the tag. Our protocol relies on a pseudorandom generator shared between readers and tags, used to derive random looking keys and encrypt secret messages. The generated keys remain indistinguishable to malicious adversaries, even after compromising the system. We present the specification of our solution using the High Level Protocol Specification Language (HLPSL) [15], a formal language to verify cryptographic protocols based on Lamport's Temporal Logic of Actions [16], and evaluate our solution with a verification tool of the Automated Validation of Internet Security Protocols and Applications (AVISPA) model checker platform [17]. The results of our evaluations show that our protocol guarantees mutual authentication of participants and strong forward secrecy of the keys in the presence of active adversaries, even when the tag is totally compromised. The results show that the protocol guarantees backward secrecy in the case of active adversaries bounded by limited communication operations.

Paper organisation. Section 2 surveys related work. Section 3 presents the security assumptions. Section 4 provides the targeted security properties we want to prove. Section 5 describes the protocol. Section 6 presents the specification of both the protocol and the security properties prior the verification. Section 7 describes the results of the automatic verification process. Section 8 provides a comparison with closest related work. Finally, Section 9 concludes the paper.

2. Related work

RFID key establishment protocols are used to rekey the internal system states in order to ensure that fresh keys are always used in every new session. The fresh keys are used in the next authentication of the tag and the reader. In our approach, the successful authentication means that the reader is authorised to access the tag additional memory. There have been several attempts to create authentication protocols for RFID systems. Most of the first security solutions based their design on the use of one way hash solutions. For instance, Weis et al. proposed in [18] a hash-lock solution to prevent unauthorised readers from reading tag contents. The design principle behind this solution includes the assumption that tags cannot be trusted to store long-term secrets when left in isolation. Thus, the secret is first sent by authorised readers to tags using a trusted environment. Then, equipped with an internal hash function, the tags perform a hash on this secret and store it within their internal memory. Then, tags enter into a locked state in which they answer to any possible query with the computed hash. Some weaknesses of the hash-lock scheme were shown in [19] leaving the protocol vulnerable to eavesdropping and replay attacks.

Another set of hash-based solutions were proposed by Ohkubo et al. [20], Avoine et al. [21] and Henrici et al. [19, 22]. In [20], the authors propose the use of hash chains to implement on-tag security mechanism requiring the tag to perform a hash operation on its ID and to send the result to the reader. The reader then makes an exhaustive search for a match. The limitations of this solution were discussed by Avoine et al. in [21] (e.g., the protocol is subject to replay attacks). They propose an enhanced hash-based RFID protocol to address privacy and authentication by introducing a time variable to prevent replay attacks. The solution of Henrici et al. [19] follows the same vein, in that the reader and tag compare hash results. A more general solution, based on hash chains, is proposed by Tsudik in [23] to guarantee mutual authentication between tags and readers. The use of chains is reported by the author as very memory consuming and the solution is qualified as prone to availability attacks on the server side. Moreover, since the publication in [24, 25] by Feldhofer et al. and Bogdanov et al., most authors in the literature agree that the use of hash functions is beyond current capabilities of low-cost RFID tags such as Gen2 tags.

Physically Unclonable Functions (PUFs) are often used as a complement to hash functions in more powerful RFID solutions [26]. Half way between traditional cryptography and physical protection defences, the PUF technology has traditionally been used to implement challenge-response protocols. PUFs originated in [27] with the conception of optical mechanisms for

the construction of physical one-way functions. The key idea is to fingerprint tags, based on their physical (i.e., fabrication) properties, e.g., delay properties of the tag circuitry or startup values of the tag memory. The main drawback is the necessity of a great number of challenges (i.e., hundreds of them) exchanged between readers and tags at a given time, to conclude the authentication process. Moreover, as it happens with many other probabilistic identification schemes [28, 29], the solution may expose the identification process between peers to an increase in response delay and power consumption, and might be acceptable only on short-distance RFID technologies with their radio spectrum in low (LF) and high (HF) frequency bands.

Bolotnyy and Robins propose in [30] a solution to complement the use of PUFs with message authentication codes, aiming to simplify the challenge-response communication scheme of previous proposals. The approach does still require the necessity of huge lists of challenge-response pairs for each PUF/tag which must be stored on back-end servers connected to the readers. Indeed, once a given pair is sent, it must not be used anymore. Otherwise, the protocol cannot guarantee that an adversary eavesdropping data will not gain advantage by performing a replay attack. Distance bounding schemes, used in protocols like [31, 32], are meant to counter this kind of attacks by handling extra delays in the exchange of messages. However, we are unaware of distance bounding protocols for UHF Gen2 RFID technologies, given their delay and power constraints.

With the motivation that traditional cryptographic protocols are too computationally intensive to be utilised, Juels et al. presented in [33] the HB+ protocol which was inherited from Hopper and Blum's early work [34]. This improved variant attempts to prevent active attackers using a statistical conjecture to bound the difficulty of learning a secret (e.g., the ID of the tag) given a sequence of randomly chosen vectors with embedded noisy information. Thus, the security relies to the difficulty of the Learning Parity with Noise (LPN) problem, which has been proved to be NP-hard. The success of the man-in-the-middle attack in [35] on HB+ leads to other versions following the basic HB+ protocol, we cite as examples, HB-MP [36], Random-HB# & HB# [37] and Trusted-HB [38].

Random-HB# & HB# have not been reported attacked yet, but the use of Toeplitz Matrix for HB# and the use of an LFSR-based Toeplitz Matrix to construct a hash function for Trusted-HB are rather expensive for constrained tags like those of Gen2 technology. HB-MP protocol was the best candidate suited to Gen2 tags, but it was explained in [37] that the amount of data transferred remains high. Furthermore, [39] reports that the protocol is vulnerable to man-in-the-middle attacks. The vulnerability found on HB-MP is due to the predictable repetition of the internal state in each session. In [40], the authors propose a solution to randomise the internal state, relying on some hash function. The solution, seems to be not yet supported by basic Gen2 tags until now. Finally, HB-PUF is a PUF version of the HB series [41]. Its main drawback is once again the requirement of storing a very large number of challenge-response pairs at either the reader or the back-end database.

Several other approaches aim at providing authentication based only on bitwise operations and very low computational primitives. The MAP family of protocols by Peris-Lopez et al. is an appropriate set of protocols characterising this class of lightweight protocols. We note LMAP [42], M2AP [43] and EMAP [44]. Their goal is to provide mutual authentication between readers and tags. The authors eliminate the use of hash primitives and involve only modular arithmetic on-tag operations. The computation of costly operations is done at the reader side. Although this effort to lighten the implementation in the tag side, none of these proposal seems to resist adversaries able to learn the secrets (including the identifier) of the tag with relative ease [45, 46, 47, 48]. For example, M2AP was broken by eavesdropping for two consecutive runs of the protocol. Improvements of these schemes [49] have also been reported as vulnerable [11]. All these examples show the necessity of formally verifying the security of new authentication schemes before their release.

Other recently efforts in [50, 51, 52, 53, 54, 55] have analysed forward security and other communication faults in RFID systems at various levels of formality. These works are the closest to our proposal (i.e., authors propose a protocol and a formal approach to verify its security). In Section 8, we analyse them more in detail and provide a comparison with our work.

3. System assumptions

3.1. Revised Security model

The security model in EPC Gen2 systems relies on the communication of one-time sequences used to encrypt sensitive data that must be sent over the insecure RFID channel. The purpose of generating one-time sequences for security is typically that both entities participating on the communications are able to repeat the sequence. However, this is not the case in the EPC Gen2 protocol, where only the tags have access to the sequence generation function. Therefore, the generated sequences cannot be reconstructed at the reader side, and must be sent as clear text over the insecure channel (i.e., tag to reader channel) [56]. To handle this security flaw, we propose to use an alternative model, and assume the use of a pseudorandom number generator (PRNG) whose algorithm is known at both sides (i.e., known by readers and tags).

3.2. Pseudorandom number generators

In our work, a PRNG is defined as a pseudorandom bit generator whose output is partitioned into blocks of a given length n. Each block defines a random-looking n-bit number said to be derived from the PRNG. The derived numbers are random-looking bits (statically independent and unbiased binary digits). The PRNG takes a single input called state (*seed*, if it is the initial state) and outputs a next state in addition to the output. All states are assumed to be hidden at all times. There are many nuances of PRNGs used in practise that are often more complicated. For example, some of them are associated to auxiliary inputs such as timestamps or counters which also can be controlled by the adversary. There have been numerous works on constructing PRNGs for symmetric encryption schemes. Common PRNGs consist of two components:

(1) a generation function that taking an internal state, generates the next output and then updates the internal state accordingly; and (2) a seed generation function that generates the initial state (and/or key) of the system.

Some designers propose a model that combines the internal state and the key of the PRNG (cf. [57] and [58]) while others separate them. Our model meets the model cited in [59] in considering the state and the *master* key separately. Indeed, the role played by the key and our concept of internal state are quite different. The key typically has a much longer lifetime and may be repeatedly used for different invocations of the PRNG. The internal state has an ephemeral nature, since it is usually updated during every iteration of the generation algorithm. Our construction concludes a solution to refresh the master key every N interactions as it will be shown in Section 5.1.

PRNGs can be based on a wide range of cryptographic primitives. The PRNGs that are in prelevant use today, are typically based on hash function or block cipher designs. Given the limited computing power of Gen2 tags, we consider in our work PRNGs built from block cipher designs with low-resource hardware constraints. Existing implementations of block cipher based designs for passive RFID tags, such as the 65nm implementation of the Advanced Encryption Standard (AES) in [60], could be adapted for our purpose with a hardware complexity of about 5000 equivalent logic gates. Other plausible solutions could be adapted from HIGHT [61], Trivium [62], Grain [63], LAMED [64], and J3Gen [65], with even lower complexity. Such designs can be adapted to implement pseudorandom permutations. A good block cipher E_k (i.e., whose permutations are dependent of a key k) is designed to approximate, as closely as possible, a random permutation function, in the sense that if the key k is not known and only input/output examples of E_k are captured, then, these should appear like input/output examples of random permutations. More information about the security and quality of block ciphers based designs is provided in Appendix A and Appendix C.

3.3. Adversary model

We assume an active adversary $\mathcal{A}$ who controls the communication channel shared between tags and sensors. Therefore, $\mathcal{A}$ can eavesdrop, store, analyse, alter, redirect, and reuse intercepted messages. $\mathcal{A}$ always knows the non-secret data and the functions that each part execute, as well as the inner working of the system (e.g., algorithms and environment associated with the protocol). Additionally, $\mathcal{A}$ can *impersonate* a sensor or a tag, and inject new messages by such controlled entities. However, $\mathcal{A}$ cannot modify those messages already sent by a non-controlled entity, nor can he prevent non-controlled entities from receiving a message already sent. Finally, $\mathcal{A}$ is motivated by any possible scenario leading to the disclosure of secret information used in the protocol. Therefore, we expect from $\mathcal{A}$ the application of the following scenarios:

- *Protocol exposure.* $\mathcal{A}$ can try to find any protocol flaw to decrypt the derived keys relying on its *a priori knowledge of the system*. Therefore, $\mathcal{A}$ can try to find any link between captured messages to correlate two or more protocol outputs. The aim is to obtain information about the derived keys.
- *Key recovering.* Using the derived keys, $\mathcal{A}$ can try to detect, at least, a couple of internal state and derived key to elaborate a relation between them. The aim is to conduct an exhaustive key search attack (cf. Appendix A.1) to derive the master key.

4. Targeted security properties

The protocol shall provide secrecy of the master and derived keys in addition to assuring that mutual authentication is done between honest participants preventing impersonation attacks. Strong notions of secrecy such as forward and backward secrecy must also be guaranteed even if adversary $\mathcal{A}$ corrupts the whole system by obtaining the session master key and the internal state of the key generation function by external means (e.g., by physically exposing the data of the tags). Therefore, our protocol shall guarantee the security properties defined below.

4.1. Mutual authentication

We define mutual authentication by the agreement of the sensor and the tag on the value of a negotiated master key in each session. When this key is also proved to be secret (i.e., nobody except the intended parties knows the key), this strong agreement excludes potential man-in-the-middle and replay attacks in which the adversary could impersonate one of the two parties.

4.2. Secrecy of the master key.

At any time period, $\mathcal{A}$ cannot recover the master key from the derived keys used in a given session and within the valid period of generation (i.e., before reaching a given threshold N).

4.3. Forward secrecy.

After the exposure of a given master key, $\mathcal{A}$ cannot compute previous master keys used in the system once the master key is refreshed. In other words, let Km_i be the i^{th} master key negotiated between the tag and the sensor, t_i be the last instant of the time interval during which Km_i is in use, and t_C be the instant of the total compromise event of the tag and the sensor. That is, if the knowledge of $\mathcal{A}$ is $\mathcal{K}_t$ at instant t after t_C then $\mathcal{A}$ cannot deduce Km_i from $\mathcal{K}_t$: $t_i < t_C < t$ then $\mathcal{K}_t \nvdash Km_i$.

4.4. Backward secrecy

After the exposure of a given master key, $\mathcal{A}$ cannot compute future master keys used in the system after the master key is refreshed. In other words, let Km_i be the i^{th} master key negotiated between the tag and the sensor, t_i be the first instant of the time interval during which Km_i is in use, and t_C be the instant of the total compromise event of the tag and the sensor. That is, if the knowledge of $\mathcal{A}$ is $\mathcal{K}_t$ at instant t after t_C then $\mathcal{A}$ cannot deduce Km_i from $\mathcal{K}_t$: $t_C < t_i < t$ then $\mathcal{K}_t \nvdash Km_i$.

5. The KEDGEN2 protocol

Our protocol assumes dynamic master key establishment based on key transportation techniques [66]. This rationale is used since parties in our system have not the same capabilities. Indeed, RFID readers are expected to have enough computational resources to calculate robust keys. Once computed, the master keys are communicated to the tags, assumed to be resource constrained components. We first present the key generation model assumed in our work. Then, we detail the protocol that uses these generated keys to encrypt secret data.

5.1. Modelling the key generation function

Let $\mathcal{BS} = (\mathcal{K}d, \mathcal{E}n, \mathcal{D}c)$ be the base symmetric encryption scheme, specified by its key generation $\mathcal{K}d$, encryption $\mathcal{E}n$ and decryption $\mathcal{D}c$ algorithms. Let $\mathcal{G}en = (\mathcal{S}, \mathcal{G})$ be the PRNG based on a block cipher primitive whose block size is the length of the derived key of the base scheme. $\mathcal{G}en$ consists of two algorithms. The first algorithm $\mathcal{S}$ is a probabilistic algorithm which takes no inputs and outputs an initial state St_1 and a master key Km_1. The second is an iterative deterministic generation algorithm $\mathcal{G}$, computing in each iteration from three inputs (a master key Km, a state St, a counter cnt) an output Kd and a new state St_i. The counter avoids cases where the same state and key are used. It is a replay defence. For $i \geqslant 1$, the generation algorithm $\mathcal{G}$ takes as input the key Km and the current state St_{i-1} (including the cnt) to generate Kd_i and St_i as: $Kd_i, St_i \leftarrow \mathcal{G}(Km, St_{i-1})$. We associate with our PRNG a re-keyed encryption scheme, which establishes a new key in every new session. The re-keying function can rely on a one way function that is responsible for changing the keys for each session.

An encryption/decryption process of the model we propose is pictured in Figure 1. The objective is to encrypt every secret message with a new derived key Kd using $\mathcal{E}n$. Thus, the derived keys are used once in each transaction while the master key Km has a longer life time.

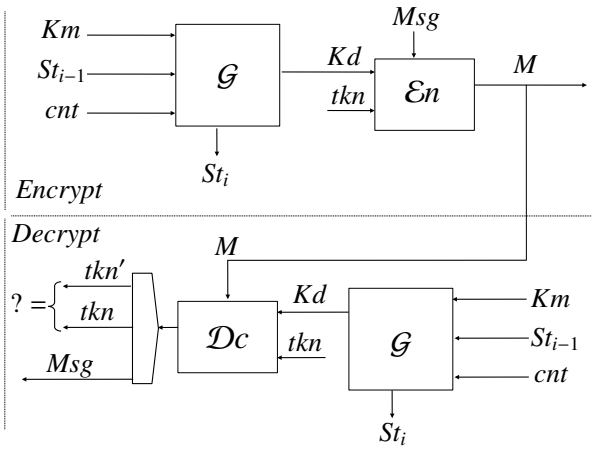

Figure 1: Proposed encryption/decryption scheme

Notice that our aim is to minimise the advantage (i.e., the likelihood) of the adversary to compromise the security of $\mathcal{G}$ using the data he recovered in each transaction. The obvious attack that takes advantage of the weaknesses of the encryption under block ciphers is the birthday attack [67]. To safely encrypt more data, a practical solution is to enlarge the limited threshold leading to birthday attacks. Thus, we can use the results of [68] by introducing a master key re-keying every $N = 2^{n/3}$ encryptions, where n is the block length. The solution increases the encryption threshold from $N = 2^{n/2}$ to $N \approx 2^{2n/3}$. This solution requires less resources than the data dependent re-keying (cf. Appendix B). In addition, it follows the basic protocol design in refreshing the keys every new session.

With this re-keying function, our encryption scheme is divided into several stages (in a same session). In stage i, all encryptions are performed using the base scheme with Km_i. An encryption counter is maintained, and when N encryptions are performed, the stage ends and a new stage begins with a new counter cnt and a new master key.

5.2. Protocol components

We assume the following components:

- *Tag.* A passive constrained device that communicates with readers via a radio interface. The tag is able to give access to its memory only with one reader at a time. It holds the function $\mathcal{G}$ of the generator $\mathcal{G}en$ and is able to derive keys according to its secret master key and state.
- *Reader.* An active entity communicating with the tags and a back-end server. It implements a radio interface to tags and a trusted interface to the server. It holds the functions $\mathcal{G}$ and $\mathcal{S}$ and is responsible for refreshing master keys when necessary.
- *Back-end server.* A trusted entity that stores in its database all tags and readers information. It is responsible for setting up the initial keys either in the tag or in the reader. It also operates to reset the system when problems arise.
- *Channels.* There are a reader-to-server channel and a reader-to-tag channel. The readers and the server are related with a high-level security channel. They are assumed to be secured with common security protocols (e.g., SSL/TLS). Reader-to-tag channel is the vulnerable channel that captures our interest.
- *Sessions.* We separate each execution of the protocol by a process named session. For each communication session between a pair of reader and tag, a different master key is established. Session key ensures the independence across sessions to avoid long-term storage of shared keys and to limit the number of ciphertexts available for cryptanalysis.

5.3. Protocol description

We describe now the steps of the protocol. For sake of simplicity, the reader and the server refer to one entity named *sensor*, since (i) readers do not store locally secret information

related to tags, and (ii) the linking channels to the server are assumed to be secure.

Each tag and sensor store a generation algorithm $\mathcal{G}$ (cf. Section 5.1) with a synchronised process. $\mathcal{G}$ is deterministic. Thus, given Km_i and $St_{i,j-1}$ in the i^{th} session and $(j-1)^{th}$ derivation, $\mathcal{G}$ always outputs the same derived key $Kd_{i,j}$ and State $St_{i,j}$. The function of initialisation $\mathcal{S}$ is performed once, (i.e., the first time the protocol is executed). It can be re-called if the system has to be set up. The sensor stores in its database all the tag information. For each tag, it records the tag pseudonym (or identifier) TagID, its current state St_i, the master keys (Km_{i-1}, Km_i) to recover the last key in case of desynchronisation, a generator counter *cnt* (cf., Section 5.1) and an encryption token *tkn*. The token *tkn* can be a counter or a timestamp. In our scheme, we are using a counter since tags are not usually connected to a server that can synchronise their clocks. We assume in all transactions that *tkn* guarantees that sent messages will be different from the ones sent in the previous transactions. It is meant to ensure the message integrity.

In case of loss in the transmission due to interference or noise, the messages are assumed to be resent with the same counters *cnt* and *tkn*. That is, if the reader or the tag does not receive the acknowledgement of the last message, the message can be retransmitted with a bit set to indicate that it is a duplicate. Hence, the receiver accepts only one validated message.

The master key Km is sent to the destination whenever the key generator $\mathcal{G}$ needs to be refreshed. This happens in the two following situations :

1. when a new session begins. In this case, the sent master key becomes the session key,
2. when the key generator counter *cnt* reaches a threshold N leading to the possible birthday attack. At this time, $\mathcal{G}$ has to be rekeyed with a new master key to extend its lifetime against the birthday attack. In this case, the sent master key replaces the actual session key.

In the following, we present the different steps of the protocol in more detail.

5.3.1. Sessions

A set-up phase is required for initialising the state and the master key Km. In this phase, authentic and secret initial keying material is distributed by a trusted third party over a secure channel.

First session. The tag and the sensor agree on an initial secret composed of: An initial master key Km, an initial internal state *seed* and a shared token *tkn*. The *cnt* of the $\mathcal{G}$ function is also initialised.

i^{th} session. In the beginning of the i^{th} session (i.e., before refreshment), the sensor and the tag share the function $\mathcal{G}$ with the same properties as those used in the $i-1^{th}$ session meaning that they use Km_{i-1} for generating derived keys. The period of generation is assumed to be still valid for unpredictable derived keys. After the establishment of the master key, the tag and the sensor share: (1) a secret master key Km_i, (2) an internal state St_i associated with a new counter *cnt* and (3) a token tkn_i which are newly refreshed.

5.3.2. Refreshing $\mathcal{G}$ in the same session

When the generation counter *cnt* reaches the value of N, the sensor sends a query for refreshing $\mathcal{G}$ as follows: Sensor sends to the tag a special command $Scmd$ xored with a new derived key $Kd_{i,j}$. Tag acknowledges the refreshment with $Kd_{i,j+1}$. Then, Sensor sends a new master key. The tag verifies the derived key and the token used to encrypt the message and in case they are equal to those it has already calculated, it accepts the master key.

5.3.3. Stages

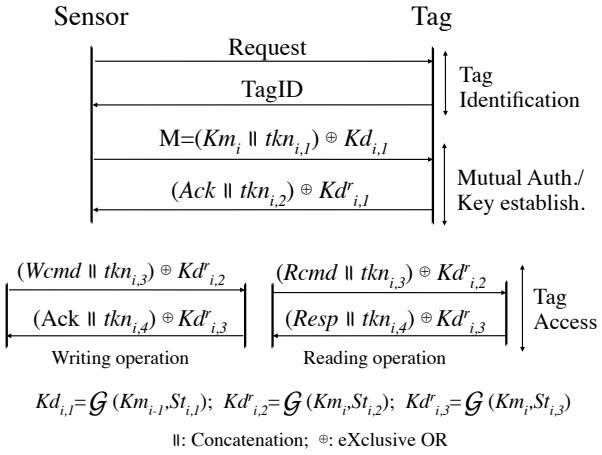

Figure 2: Main stages of the protocol

In each new session, three stages are required. One stage for identification and another for authentication and key establishment. The third stage is a consequence of the successful authentication which means the access to the internal memory of the tag. These stages are shown in Figure 2.

1. Tag identification stage. The sensor starts by sending successive requests to the Tag until it obtains the tag's pseudonym TagID. The sensor checks in its database the received TagID. If there is a match, the sensor associates the TagID with the EPC identification and the related secret information (i.e., the master key and the state previously negotiated). Both sides must have the same secrets. Otherwise, the next authentication process will fail. The sensor calculates the response including the derived key (by using $\mathcal{G}$ with the valid previous master key Km_{i-1} and St_{i-1}) to prove that it recognises Tag and *xor*es the result with the new established master key Km_i. Notice that the sensor stores both the current and previous master keys to handle desynchronisation.

2. Mutual authentication and refreshment stage. Upon receiving the message M, Tag checks the derived key used for encryption. Then, it calculates a new $Kd'_{i,1}$ (i for current session) and decrypts M by applying an *xor* operation as follow: $Op = (Km_i, tkn_{i,1}) \oplus Kd_{i,1} \oplus Kd'_{i,1}$. If the decrypted suffix of Op is equal to the predefined token $tkn_{i,1}$, then Tag authenticates Sensor and accepts Km_i as a new master key. It returns an acknowledgement Ack associated with a new derived key $Kd^r_{i,1}$

(r for refreshed key) set from the refreshed values. Otherwise, Tag does not accept the sensor's key and aborts the communication. Upon receiving the value of $Kd^r_{i,1}$, Sensor verifies it and authenticates Tag, in case of validity.

3. Tag access stage. After a successful authentication, the sensor is authorised to access the tag. Thus, it has the ability to execute privileged commands like reading or writing on it. The same process of authentication is used to perform an access operation. Instead of sending the master key, the sensor sends the data to be written on the tag or the tag sends the data required by the sensor encrypted with a fresh derived key:

- Writing operation: Sensor begins by sending the write command *Wcmd* concatenated with the token and *xor*ed with a new $Kd^r_{i,2}$. Tag verifies this key and accepts the command if the value is valid. Then, it acknowledges the reception with $Kd^r_{i,3}$. Otherwise, Tag aborts the communication.
- Reading operation: Sensor begins by sending the read command *Rcmd xor*ed with the new generated key $Kd^r_{i,2}$. If Tag accepts the request by checking $Kd^r_{i,2}$, it sends the response $Resp$ *xor*ed with $Kd^r_{i,3}$.

5.3.4. Concurrent executions

The Gen2 technology is highly concurrent as a large number of tags could be interrogated at the same time. Thus, it is important for participants to separate concurrent protocol executions. This issue is usually handled by adding a session ID field to the exchanged messages. In our protocol, we assume that each protocol session is associated with an initial internal state, a secret *Km*, and a token *tkn* that differentiate all the tags in the system. The sensor can run concurrently many protocol sessions at a time since it maintains a set of tag information. In contrast, the tag cannot run concurrently many protocol sessions at a time, particularly when it needs to update its secrets simultaneously (e.g., the secrets have to be updated before starting a new session). In the sequel, we consider that the tag can respond to several identification requests by sending its pseudonym but for privileged requests (reading/writing accesses), the tag does not respond to simultaneous queries, nor is it able to increment its internal state and token two times simultaneously. Finally, for synchronisation reasons, the tag has to run each session for a small period of time, and then switches off automatically (even if the session has not ended).

6. Verifying the security of the protocol

We use model checking techniques to verify whether a security property holds in a finite state machine. The goal is to find errors (e.g., logical flaws) and attacks against the protocol implementation in accordance with the security assumptions provided in Section 3. Automated reasoning is highly desirable to avoid errors associated with hand-written proofs. If a security design of complex systems is verified successfully by an automated tool, it increases the confidence of the system users.

Let us begin by presenting the tool we use for the verification. Then, we give some notions about the HLPSL input language and the structure of the outputs.

6.1. Checking Tool

There is a number of successful protocol verification tools that are supporting algebraic reasoning, e.g., the extended ProVerif [69], Maude-NRL Protocol Analyzer (MaudeNPA) [70], On the Fly Model Checker (OFMC) [71] and Constraint-Logic based Attack Searcher (CL-AtSe) [72]. They use different models and proof techniques. For example, extended ProVerif is based on tree automata and Horn clauses techniques. MaudeNPA is based on rewriting techniques and backward search of *bad* states. OFMC is based on a state space exploration and CL-AtSe is based on a constraint solving technique. Each tool has some strengths and weaknesses [73]. As a candidate, we use CL-AtSe protocol analyser, the probably most mature tool using the constraint solving technique [74]. The tool was part of the AVISPA project [17] that has been extended recently by the Avantssar [75] project. In this paper we use the new version of CL-AtSe [72] to verify our protocol. The AVISPA platform is a suite of applications commonly used for formal specification and automated validation and verification of cryptographic protocols. It is composed of several modules: A translator called HLPSL2IF [76] is used to transform a given HLPSL specification to a low level specification IF and four different verification tools (CL-AtSe, OFMC, SAT based Model-Checker (SAT-MC) [77] and Tree Automata based Protocol Analyser (TA4SP) [78]) to analyse the IF specifications. The choice of CL-AtSe tool to verify our protocol refers to its ability to analyse protocols taking in account the algebraic properties of operators like xor or exponentiation in addition to the possibility to run many consecutive sessions.

The CL-AtSe tool. Constraint-Logic based Attack Searcher consists in running the protocol or set of services in all possible ways by representing families of traces with positive or negative constraints on the intruder knowledge, on variable values, on sets, etc. Thus, each run of a service step consists in (1) adding new constraints on the current intruder and environment state, (2) reducing these constraints down to a normalised form for which satisfiability is easily decidable, and (3) deciding whether some security property has been violated up to this point. CL-AtSe does not limit the service in any way except for bounding the maximal number of times a service can be iterated, in the case such an iteration (or loop) is specified by the user. Otherwise, the analysis might be non-terminating on secure services and only heuristics, approximations, or restrictions on the input language could lift this limitation. In our protocol verification, we specify three consecutive iterations (i.e., sessions). This number is indeed representative of different steps of our protocol and is sufficient to check the properties we wish to verify.

6.2. HLPSL format

The protocol and the security properties are specified in the High Level Protocol Specification Language (HLPSL) [15].

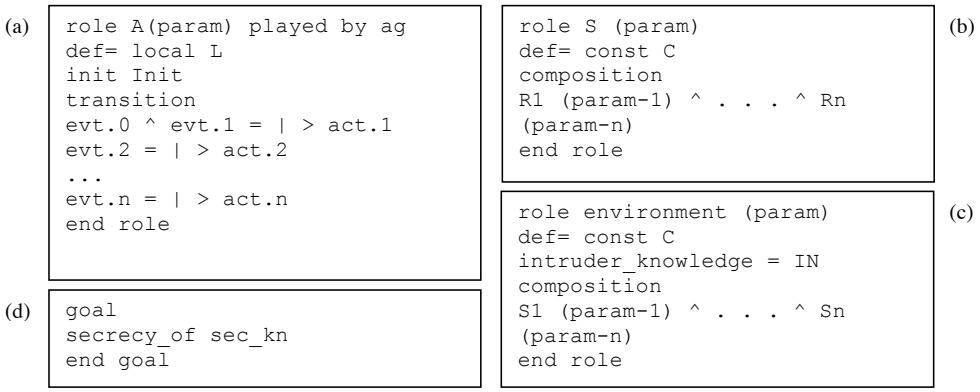

Figure 3: HLPSL main elements. (a) Basic role structure. (b) Session role structure. (c) Environment role structure. (d) Secrecy in the goal section.

HLPSL is a specification language for formalising protocols and security goals based on Lamport's Temporal Logic of Actions (TLA) [16]). The language, developed in the context of the AVISPA project [17], is a role-based language. Roles can be `basic` (e.g., agent roles) describing the action of a legitimate participant during the execution of the protocol or `composed` (e.g., session and environment roles) describing scenarios of basic roles to model an entire protocol run. Finally, the HLPSL language allows to specify the knowledge and capacities of the adversary model.

Basic roles. Figure 3(a) shows how the basic role is generally structured. Each basic role declares its name (`A`), its initial information or parameters (`param`) and the agent playing the role (`ag`). The basic role can declare a set of local variables (`L`). The `init` section assigns the initial values to the local variables, if required.

The `transition` section describes changes of the agent state. It consists of a trigger (e.g., `evt.2`) and an action (e.g., `act.2`) to be performed when the trigger event occurs. The `= | >` symbol separates the two phases.

Composed roles. Composed roles combine basic roles, either in parallel or in sequence. HLPSL defines two composed roles: the session role and the environment role. Actions, in composed roles, are not defined in a `transition` section like in basic roles. Rather, a `composition` section is defined to instantiate other roles `Ri` or `Si`, with sets of parameters `param-i`, that run in parallel (cf. Figures 3(b) and 3(c)). The session role, referred by `S` in Figure 3(b), instantiates in its composition section the basic roles and the different channels relating them while the environment role instantiates in its composition section all the sessions to be run (`Si`). The environment role is called the main role, as it declares the global constants (`C`) and defines the intruder knowledge denoted by `IN`.

Security properties. HLPSL provides an independent section to declare the security properties required, named `goal`. The goal declaration can be done either by using predefined macros of the predefined security properties (secrecy, weak authentication, strong authentication) or by using Linear Temporal Logic formulas [16]. We are interested in the predefined secrecy and strong authentication properties. We use the predefined secrecy property to check whether the secrecy of the key is maintained in a given session and to check (with a slight change of the specification) whether the forward/backward secrecy defined in Sections 4.3 and 4.4 are guaranteed in last/next session respectively, after the compromising of the system in a given session. We use also the authentication property to validate the goals defined in Section 4.1.

- Secrecy is modelled by means of the goal predicate *secret(Km,sec_km,Sensor,Tag)* standing for the value of term *Km* is a secret shared only between agents *Sensor* and *Tag*. The secrecy property is violated every time the adversary learns a value that is considered as secret and that he is not allowed to know (i.e., *Km*).

- Authentication is modelled by means of the goal predicates $witness(A, B, id, T1)$, $request(B, A, id, T1)$ and $wrequest(B, A, id, T1)$. These predicates are used to check if an instance of a role is right in believing that its peer is present in the current session. It is done by agreeing on a certain value (e.g., $T1$) which is typically fresh. The predicates always appear in pair and have the same third parameter. This third parameter *id* is the identifier of the authentication goal and it is used in the goal section of the HLPSL code. There exists two definitions of authentication: weak and strong authentication.

  1. $witness(A, B, id, T1)$ for a strong or weak authentication properties of A by B on $T1$, declares that agent A is witness for information $T1$. This goal will be identified by the constant *id* in the goal section;
  2. $request(B, A, id, T1)$ for a strong authentication property of A by B on $T1$, declares that agent B requests a check of the value $T1$. This goal will be identified by the constant *id* in the goal section;
  3. $wrequest(B, A, id, T1)$ similar to request, but for a weak authentication property. It is used to specify an authentication goal with no replay protection.

  Strong authentication is an extension of the weak authentication which precludes replay attacks. We can thus conclude that, if strong authentication is achieved, then $T1$ has not been previously received by B in a given session.

```
role sensor(...)
…
/\ witness (A,Tag',sensor_tag_kd1,
   keygen(KM',succ(KM',InState')))
…
end role

role tag(...)
…
/\request
(Tag,Sensor,sensor_tag_kd1,keygen
(KM.InState))

end role
```

```
role environment(...)
…
tag_sensor_kd1 :protocol_id
…
end role

goal

  authentication_on
sensor_tag_kd1

end goal
```

Figure 4: Strong authentication property definition

Each property is added to the honest role and to the goal section. It is identified by `protocol_id` type. Figure 4 shows a declaration of a strong authentication property of the *sensor* by the *tag* on the value of $Kd_1 = keygen(KM', succ(KM', InState'))$ declaring that agent *sensor* is witness for the value of Kd_1 and that agent *tag* requests a check of the value Kd_1. This goal is identified by the constant *sensor_tag_kd*1 in the `goal` section.

6.3. Output format

After the verification process, the output describes the results, and under what conditions they have been obtained (e.g., Figure 7 shows the verification results of the KEDGEN2 protocol). The output format is nearly common to all tools of the framework. In the `SUMMARY` section, it indicates if the protocol is safe, unsafe, or if the analysis is inconclusive. In a second section titled `DETAILS`, the output shows conditions under what the protocol is declared safe/unsafe/inconclusive. If a security property of the input specification is violated then the tools output a warning, some details about the analysis (e.g., whether the considered model is typed or untyped), the property that was violated (e.g., authentication), statistics on the number of explored states, and, finally, an `ATTACK TRACE` that gives a detailed account of the attack scenario. If no attack was found, then similar information is provided without announcing any violation and attack trace.

6.4. Our HLPSL specification

The specification of both the protocol and the security goals is described into four HLPSL sections: the sensor, the tag, the environment roles and the goal. Figure 5 shows the specification with mutual authentication and secrecy of the master key goals. The generation function $\mathcal{G}$ is specified by two functions *keygen* and *succ*. The first function generates the derived keys and the second one generates accordingly the new state (i.e., $InState$). Figure 6 shows the specification of the protocol to handle the forward secrecy. Note that for the cases of forward and backward secrecy, we have slightly changed the specification (compared to Figure 5) as the AVISPA tool only supports a single execution trace. Thus, we have modelled the execution of two consecutive iterations in order to show whether leaking a secret during session *i* helps the adversary to obtain secrets from session *i-1* for forward secrecy or session *i+1* for backward secrecy. In the following, we detail the specification and evaluation results.

7. Evaluation results

For each security property defined in Section 4 and specified in Section 6, we show in this section the results obtained after the evaluation of our protocol specifications under the CL-AtSe tool.

7.1. Mutual authentication

Figure 7a shows the results of the evaluation of the mutual authentication property. To obtain these results, we specify an iteration of the protocol with legitimate roles and give to the adversary the knowledge of the generation functions, roles and standard commands used in the KEDGEN2 protocol communication (cf., Figure 5). In the HLPSL language, the authentication property is specified using the *witness*/*request* predicates. These predicates are used to check if an instance of a role is right in believing that its peer is present in the current session. We use the HLPSL strong authentication definition to require that a given value is accepted by the sensor in exactly the same session in which it was proposed by the tag. We add these predicates to the tag and sensor transactions to evaluate the authentication of each of the two roles and prevent man-in-the-middle and replay attacks. The tool finds no attack violation of the strong authentication property. This strong property guarantees the resilience to man-in-the-middle and replay attacks in which the adversary could impersonate one of the two parties.

7.2. Secrecy of the master key

Figure 7a shows the results of the secrecy property evaluation. We recall that the secrecy of the master key when shared securely between the tag and the sensor is mathematically maintained since the security threshold N of distinguishability is not reached. In other words, the adversary is not able to detect correlations between the outputs of $\mathcal{G}$, named the derived keys. The model checker is used in our evaluation to confirm that the adversary is not able to desynchronise the two participants and replay some messages to reconstruct the master key (and with that, the secret messages encrypted using such a key). To verify the secrecy of the master key, we specify with HLPSL a single instance of the protocol with legitimate roles and give the adversary the knowledge of inner working of the system (cf., Figure 5). Secrecy is modelled by the mean of the goal predicate $secret(Km, sec\_km, Sensor, Tag)$ standing for the value of term Km is a secret shared only between agents Sensor and Tag. The secrecy property is violated every time the adversary learns the value Km that is considered as secret.

7.3. Forward secrecy

Figures 7b and 7c show the forward secrecy evaluation results. To prove forward secrecy, we consider a setting in which the tag and the sensor try to establish a new master key $NewKm$ using the previous master key Km. Once $NewKm$ has been established, we reveal to the adversary the internal states $NewKm$, $InState$, and Tkn of both the tag and the sensor. Our goal is to prove that this knowledge is not sufficient to enable the adversary to compute the previous Km. We prove first that

```
role sensor(A: agent,
  DataBase: (agent.text.message.text) set,
  Snd,Rcv : channel (dy)) played_by A def=

  local
    Tag:agent, InState:message,
    Tkn,KM,NewKM: text, State:nat

  init
    State := 0

  transition

    0.State=0 /\ Rcv(start) =|>
      Snd(A.reqID) /\ State':= 1

    1.State=1 /\ Rcv(Tag')
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    State':=0 /\ NewKM':= new()
    /\ DataBase':=cons(Tag'.Tkn'.
     succ(KM',InState').KM',
     delete(Tag'.Tkn'.
     InState'.KM',DataBase))
    /\ Snd(xor(NewKM'.Tkn',
     keygen(KM',succ(KM',InState'))))
    /\ State':=2
    /\ witness (A,Tag',sensor_tag_kd1,
    keygen(KM',succ(KM',InState')))
    /\ secret(KM',sec_km1,{A,Tag'})

    2.State=2
    /\ Rcv(keygen(KM',InState'))
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    request(A,Tag,tag_sensor_kd1r,
    keygen(KM'.InState))

end role
```

```
role tag(Tag,Sensor: agent,
  InState : message,
  KM : text,
  Tkn:text,
  Snd,Rcv: channel(dy)) played_by Tag def=

  local
     State : nat

  init
     State := 0

  transition

     0.State=0 /\ Rcv(Sensor.reqID) =|>
     State':=1 /\ Snd(Tag)
     /\ InState':= succ(KM,InState)

     1.State=1 /\ Rcv(xor((KM'.Tkn),
     keygen(KM.InState))) =|>
     Snd(Tag.keygen(KM'.succ(KM'.InState)))
     /\InState':=succ(KM'.succ(KM'.InState))
     /\ State':=0
     /\request(Tag,Sensor,sensor_tag_kd1,
       keygen(KM.InState))
     /\witness(Tag,Sensor,tag_sensor_kd1r,
       keygen(KM'.succ(KM'.InState)))

end role
```

```
role environment() def=

  const
    sensor,tag1,tag2: agent,
     token1,token2: text,
     instate1,instate2: message,
     km1,km2:text,
     reqID: text,
     succ,keygen: function,
     r2t,t2r: channel (dy),
      a:agent,
     sec_km1,sensor_tag_kd1,
     tag_sensor_kd1r : protocol_id


  intruder_knowledge={reqID,succ,
   keygen,sensor,tag1,tag2,keygen}

  composition
    reader(sensor,
       {
          tag1.token1.instate1.km1,
          tag2.token2.instate2.km2
       },
       r2t,t2r)
     /\ tag(tag1,sensor,instate1,
        km1,token1,t2r,r2t)
     /\ tag(tag2,sensor,instate2,
        km2,token2,t2r,r2t)

end role
```

```
goal
    secrecy_of sec_km1
    authentication_on sensor_tag_kd1
    authentication_on tag_sensor_kd1r
end goal


environment()
```

Figure 5: HLPSL specification of KEDGEN2 protocol to check for strong authentication and secrecy.

```
role sensor(A: agent,
  DataBase: (agent.text.message.text) set,
  Snd,Rcv : channel (dy)) played_by A def=

   local
     Iter:nat,Tag:agent,
     InState:message,Tkn,KM,NewKM: text,
     State:nat

   init
    State := 0

   transition

    0.State = 0 /\ Rcv(start) =|>
    Snd(A.reqID) /\ State':= 1

    1.State = 1 /\ Rcv(Tag')
    /\ in(Tag'.Tkn'.InState'.KM',DataBase)
    =|>
    State':=0 /\ NewKM':= new()
    /\ Snd(Tag'.NewKM'.Tkn'.
    succ(NewKM',succ(KM',InState')).
    xor(NewKM'.Tkn',
    keygen(KM',succ(KM',InState'))))
    /\ DataBase' := cons(Tag'.Tkn'.
    succ(NewKM',succ(KM',
    succ(KM',InState'))).NewKM',
    delete(Tag'.Tkn'.InState'.KM',
    DataBase))

    /\ secret(KM',sec_km1,{A,Tag'})


end role
```

```
role tag(Tag: agent,
  InState : message, % InState = instate0
  KM : text,Tkn:text,
  Snd,Rcv: channel(dy)) played_by Tag def=

   local
     Reader: agent,
     State : nat

   init
     State := 0

   transition

     0.State=0 /\ Rcv(Reader'.reqID) =|>
     State':=1 /\ Snd(Tag)
     /\ InState' := succ(KM,InState)

     1.State=1 /\ Rcv(xor((KM'.Tkn),
     keygen(KM,InState))) =|>
     State':=0
     /\ Snd(Tag.keygen(KM',
     succ(KM',InState)))
     /\ InState' :=
     succ(KM',succ(KM,InState))


end role
```

```
role environment() def=

     const
       sensor,tag1,tag2: agent,
       token1,token2: text,
       instate1,instate2: message,
       km1,km2:text,
       reqID: text,
       succ,keygen: function,
       r2t,t2r: channel (dy),
       sec_km1, sec_resp,
       sensor_tag_kd0,
       tag_sensor_kd1 : protocol_id


   intruder_knowledge={reqID,succ,keygen
   ,sensor,tag1,tag2,keygen}

     composition
     reader(sensor,
       {
          tag1.token1.instate1.km1
          ,tag2.token2.instate2.km2
       },
       r2t,t2r)
     /\ tag(tag1,instate1,km1,
     token1,t2r,r2t)
     /\ tag(tag2,instate2,
        km2,token2,t2r,r2t)
   end role
```

```
goal
    secrecy_of sec_km1
end goal


environment()
```

Figure 6: HLPSL modified specification of KEDGEN2 protocol to check for forward secrecy.

```
INPUT V7-1-ProtocolAuthentifSecrecy.if
SUMMARY NO_ATTACK_FOUND
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION
2.5-8_(February_23th_2011)
STATISTICS TIME 44 ms
TESTED 105 transitions
REACHED 34 states
READING 0.04 seconds
ANALYSE 0.00 seconds
```

(a) Authentication and secrecy evaluation

```
INPUT V7-6-forward-orig-chiff.if
SUMMARY ATTACK_FOUND
GOAL:secrecy_of_sec_km1(km1,set_53)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION 2.5-8
_(February_23th_2011)
STATISTICS TIME 28 ms
TESTED 10 transitions
REACHED 6 states
READING 0.01 seconds
ANALYSE 0.02 seconds
```

(b) Forward secrecy evaluation (original specification)

```
INPUT V8-forward-chiff.if
SUMMARY NO_ATTACK_FOUND
GOAL:secrecy_of_sec_km1(km1,set_53)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION 2.5-8
_(February_23th_2011)
STATISTICS TIME 24 ms
TESTED 27 transitions
REACHED 17 states
READING 0.01 seconds
ANALYSE 0.01 seconds
```

(c) Forward secrecy evaluation (modified specification)

```
INPUT V7-6-backward-chiff.if
SUMMARY ATTACK_FOUND
GOAL: secrecy_of_sec_km1(n3(NewKM),set_55)
DETAILS TYPED_MODEL
BACKEND CL-ATSE VERSION
2.5-8_(February_23th_2011)
STATISTICS TIME 928 ms
TESTED 16 transitions
REACHED 12 states
READING 0.05 seconds
ANALYSE 0.88 seconds
```

(d) Backward secrecy evaluation

Figure 7: Evaluation results.

the original specification of our protocol (cf. Figure 5) does not provide forward secrecy. This is shown with the results in Figure 7b. The analysis of the attack trace shows that after establishing and sending the new master key to the tag (i.e., $M = (NewKm||Tkn) \oplus Kd_1$ where $Kd_1 = \mathcal{G}(Km, InState_1)$), the adversary obtains Km in the next generation of $InState$ ($InState_2 = \mathcal{G}(NewKm, InState_1)$) relying on the knowledge of $NewKm$ and Kd_1. The countermeasure is to hide the generation of $InState_2$ by values which are not deduced by the adversary. This way, the adversary cannot obtain the key Km. In fact, by changing $\mathcal{G}(NewKm, InState_1)$ to $\mathcal{G}(NewKm, \mathcal{G}(Km, InState_1))$, we use a double generation of the initial state depending on values that cannot be computed by the adversary (i.e., Km). This modification is shown in Figure 6. The evaluation results in Figure 7c show that the modified version satisfies the forward secrecy property even under the hypothesis of a complete compromise in the following sessions.

7.4. Backward secrecy

Figure 7d shows the results of the backward secrecy property evaluation. We consider two executions of the sensor. One execution in which the tag and the sensor establish a master key Km and another one where we reveal to the adversary the last secrets (i.e., $Km, State, Tkn$) of both the tag and the sensor. The goal is to verify if this knowledge is sufficient to enable the adversary to compute the new master key $NewKm$ related to this execution. The results show that the protocol is insecure. CL-AtSe finds an attack on the secrecy of the new master key. Indeed, if the adversary follows all the messages sent in the network, it is possible to reconstruct the following master key $NewKm$ because the new derived key used to encrypt the message of refreshment (i.e., 3^{rd} pass in the Figure 2) can be computed. The new derived keys are based on the previous secrets that the adversary has already gained, and once obtaining these secrets, the adversary takes all the power of the target tag itself. He can trace it at least during the authentication immediately following the attack. This attack can be avoided by changing the adversary capacities. If the adversary does not eavesdrop on the tag continuously after the time of corruption, i.e., missing the master key establishment transaction, then it will not be possible to predict the next refreshed derived keys. This notion is known to *restricted* backward security through key insulation [79, 80]. This assumption is realistic since in typical RFID system environments, tags and readers operate only at a short communication range and for a short period of time.

8. Comparative discussion

In this section, we present the most relevant works related to our proposal as well as a comparative discussion between them.

Several papers e.g., [50, 51, 52, 53, 54, 55] have analysed forward security and other communication faults in RFID systems at various levels of formality. Some of them define the authentication and secrecy (named also privacy) in computational model, typically in terms of games. For example, in [50], the authors define two security protocols to assure authentication and forward secrecy using the universal composability framework. After detecting a synchronisation problem related to [50], a new series of protocols was proposed by the same authors. The last version in [51] ameliorates the protocol and includes a verification of the backward secrecy property using the same framework. The authors in [52], use a game-based approach to prove the robustness of an RFID protocol against

Table 1: Comparison between recent related work using various models of formality

Main characteristics	[50]	[51]	[52]	[53]	[54]	[55]	KEDGEN2
Formal model	Computational	Computational	Computational	Symbolic	Symbolic	Symbolic	Symbolic
Framework	Universal composability	Universal composability	CryptoVerif	ProVerif	FDR	AVISPA (OFMC)	AVISPA (CL-AtSe)
Forward secrecy	√	√	√	√	–[1]	–	√
Backward secrecy	–	√[2]	–	–	–	–	√[3]
Authentication	√	√	√	√	√	√	√
Cryptographic primitives	Pseudorandom generator. e.g., shrinking generator	PRNG[4]	Three distinct hash functions	Two distinct hash functions	Hash function + PRNG	Hash function	PRNG
Application on highly constrained tags	Possible	Possible	Not possible	Not possible	Not possible	Not possible	Possible

√: Checked by the authors – not checked by the authors
[1] A different definition of forward secrecy is checked named forward untraceability
[2] Checked under the same adversary used for verifying the forward secrecy
[3] Checked under a weaker adversary used for verifying the forward secrecy
[4] PRNG: Pseudorandom Number Generator

a man-in-the-middle adversary. They do not propose a new protocol to be applied on constrained tags but a new method to prove the security of the OSK protocol [81] that they combine with a mechanism to synchronise the internal state of the tag and reader. The resulted protocol can be applied on RFID tags supporting hash functions. The security of the protocol is proved using the computational model of CryptoVerif verification tool combined with some handwritten proofs to overcome the limitations of the tool regarding desynchronisation and forward privacy verification.

Works in [53], [54] and [55] have used the symbolic model to formally verify the security properties. These works are closest to ours. The advantage of using the symbolic model, as our work does, is its ease to automatically prove the security of cryptographic protocols and to clarify these complex protocols with provided definitions of formal languages. In view of this, the authors in [53] use the applied pi calculus language with the ProVerif automated verification tool and apply their proposed techniques to the OSK protocol [81] in order to formally prove the untraceability and forward privacy properties. The proposed technique, which consists in the concept of frame independence between sessions, meet our security goals. However, the proposed verification technique is applied only on one class of protocols that the authors call "single step" identification. Furthermore, this new technique is applied on protocols with two distinct hash functions. This is only possible on tags computationally strong enough to use such functions. These two criteria make the solution different from our proposal, i.e., our proposal uses more steps for both identification and authentication in the context of Gen2 tags (without hashing capabilities). In [54], the authors use an automated verification tool called FDR (Failure Divergence Refinement). The work can be compared to ours as it also uses a model checking tool to verify the secrecy and authentication of an RFID protocol. However, the use of a hash based scheme added to a pseudorandom number generator to implement the protocol presents a different solution model. Moreover, as opposed to our proposal, authors do not consider in their work strong secrecy notions such as forward and backward secrecy that handle linkability between the sessions. In [55], the authors propose a new protocol that assures mutual authentication and privacy which they define as anonymity and forward untraceability. The verified property of forward untraceability is different from forward secrecy. Authors define the protocol as attacked when the attacker detects twice the same hash result, which means detecting the same tag. Whereas in our case, an attack is shown when an attacker obtains the secret keys of last sessions of communicated keys for a given tag. Furthermore, the authors propose a one-way hash function as a solution for a secured RFID protocol and use the AVISPA OFMC tool for automated verification.

Differently from all cited protocols achieving a formal verification of their proposals in the symbolic model, our proposed KEDGEN2 protocol is applied on highly constrained tags such as Gen2 since it is only based on a pseudorandom number generators. Recall that in our work, we assume that the use of hash functions is beyond the capabilities of low-cost RFID tags, as reported in [24, 25] for EPC Gen2 tags. KEDGEN2 achieves mutual authentication, and forward and backward secrecy in different conditions. We prove these properties in the AVISPA framework using the Constraint-Logic based Attack Searcher

(CL-AtSe) automated verification tool. This tool is based on an attack construction methodology attempting to find vulnerabilities using algebraic properties of protocols. This approach is different from proof construction methodologies. Proof construction methodologies are suitable for proving correctness and completeness rather than finding vulnerabilities in a security protocol.

It is worth noting that in [50] and [51], the authors apply their protocols on the same category of Gen2 tags. However, they use the universal composability framework, which is based on the computational model.

To summarise, Table 1 shows the different aspects that differentiate our KEDGEN2 protocol to such previous efforts.

9. Conclusion

With the goal of addressing security concerns on the use of the EPC Gen2 RFID technology in sensitive domains, we presented the specification and verification of a key establishment and derivation protocol for Gen2 systems. The KEDGEN2 protocol achieves secure data exchange between tags and readers, based on a key generation model adapted to Gen2 RFID tags. The generated keys are used in the proposed protocol as one time encryption keys. To guarantee the security of the protocol, the generation function has to respond to a number of properties, including the resilience against key recovery and the indistinguishability of the derived keys. We described the steps of our protocol and verified the expected properties under the presence of an active adversary. The current version of the protocol guarantees the properties of mutual authentication and forward secrecy. Backward secrecy is also verified under weaker adversary assumptions (consistent with typical RFID environments). For future work, we plan to study how to *k*-anonymise information stored in the tag memory. The goal is to reduce the privacy impact on sensitive information transmitted without protection in the insecure tag-to-sensor channel. We also plan to conduct SPICE simulations to evaluate the power-consumption characteristics of our solution.

Acknowledgements: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. J. Garcia-Alfaro acknowledges the support received from the Spanish Government projects TSI2007-65406-C03-03 E-AEGIS, TIN2011-27076-C03-02 CO-PRIVACY, CONSOLIDER INGENIO 2010 CSD2007-0004 ARES, and TIN2010-15764 N-KHRONOUS.

References

[1] Q. Z. Sheng, S. Zeadally, A. Mitrokotsa, Z. Maamar, RFID technology, systems, and applications, J. Network and Computer Applications 34 (3) (2011) 797–798.

[2] W. Jakkhupan, S. Arch-int, Y. Li, Business process analysis and simulation for the RFID and EPCglobal Network enabled supply chain: A proof-of-concept approach, J. Network and Computer Applications 34 (3) (2011) 949–957.

[3] F. Rizzo, M. Barboni, L. Faggion, G. Azzalin, M. Sironi, Improved security for commercial container transports using an innovative active RFID system, J. Network and Computer Applications 34 (3) (2011) 846–852.

[4] W. Yao, C.-H. Chu, Z. Li, Leveraging complex event processing for smart hospitals using RFID, J. Network and Computer Applications 34 (3) (2011) 799–810.

[5] P. Najera, J. Lopez, R. Roman, Real-time location and inpatient care systems based on passive RFID, J. Network and Computer Applications 34 (3) (2011) 980–989.

[6] EPCglobal., EPC Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860-960 MHz, Tech. rep., Version 1.2.0 (2008).
URL `http://www.epcglobalinc.org/standards/`

[7] P. Manzanares-Lopez, J. P. Muñoz-Gea, J. Malgosa-Sanahuja, J. C. Sanchez-Aarnoutse, An efficient distributed discovery service for EPCglobal network in nested package scenarios, J. Network and Computer Applications 34 (3) (2011) 925–937.

[8] W. Tounsi, J. Garcia-Alfaro, N. Cuppens-Boulahia, F. Cuppens, Securing the communications of home health care systems based on RFID sensor networks, in: Eighth Annual Communication Networks and Services Research Conference, IEEE, 2011, pp. 284–291.

[9] J. Melia-Segui, J. Garcia-Alfaro, J. Herrera-Joancomarti, Analysis and improvement of a pseudorandom number generator for EPC Gen2 tags, Financial cryptography and data security (2010) 34–46.

[10] J. Melia-Segui, J. Garcia-Alfaro, J. Herrera-Joancomarti, A practical implementation attack on weak pseudorandom number generator designs for EPC Gen2 tags, Wireless Personal Communications 59 (1) (2011) 27–42.

[11] T. Cao, E. Bertino, H. Lei, Security analysis of the SASI protocol, IEEE Transactions on Dependable and Secure Computing 6 (2009) 73–77.

[12] T. Li, R. Deng, Vulnerability analysis of EMAP-an efficient RFID mutual authentication protocol, in: The Second International Conference on Availability, Reliability and Security, ARES'07, IEEE, 2007, pp. 238–245.

[13] T. Li, G. Wang, Security analysis of two ultra-lightweight RFID authentication protocols, New Approaches for Security, Privacy and Trust in Complex Environments–IFIP-SEC'07 232 (2007) 109–120.

[14] W. Tounsi, N. Cuppens-Boulahia, F. Cuppens, J. Garcia-Alfaro, Formal Verification of a Key Establishment Protocol for EPC Gen2 RFID Systems: Work in Progress, in: J. Garcia-Alfaro, P. Lafourcade (Eds.), Foundations and Practice of Security, Vol. 6888 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 242–251.

[15] Y. Chevalier, L. Compagna, J. Cuellar, P. Drielsma, J. Mantovani, S. Moedersheim, L. Vigneron, A high level protocol specification language for industrial security-sensitive protocols, in: Proceedings of Workshop on Specification and Automated Processing of Security Requirements, Vol. 180 of (SAPS'04), 2004.

[16] L. Lamport, The temporal logic of actions, ACM Transactions on Programming Languages and Systems–TOPLAS'94 16 (1994) 872–923.

[17] A. Armando, D. Basin, Y. Boichut, Y. Chevalier, L. Compagna, J. Cuéllar, P. Drielsma, P. Heám, O. Kouchnarenko, J. Mantovani, et al., The AVISPA tool for the automated validation of internet security protocols and applications, in: Proceedings of the 17th International Conference on Computer Aided Verification, (CAV'05), Springer, 2005, pp. 135–165.

[18] S. Weis, S. Sarma, R. Rivest, D. Engels, Security and Privacy Aspects of Low-Cost Radio Frequency Identification Systems, in: International Conference on Security in Pervasive Computing, SPC'03, 2003.

[19] D. Henrici, P. Müller, Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers, in: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, IEEE, 2004, pp. 149–153.

[20] M. Ohkubo, K. Suzuki, S. Kinoshita, Efficient Hash-Chain Based RFID Privacy Protection Scheme, in: International Conference on Ubiquitous Computing – Ubicomp, Workshop Privacy: Current Status and Future Directions, 2004.

[21] G. Avoine, P. Oechslin, A Scalable and Provably Secure Hash Based RFID Protocol, in: International Workshop on Pervasive Computing and Communication Security, PerSec'05, IEEE, 2005.

[22] D. Henrici, P. Müller, Providing Security and Privacy in RFID Systems Using Triggered Hash Chains, in: Sixth Annual IEEE International Conference on Pervasive Computing and Communications, PerCom'08, 2008.

[23] G. Tsudik, YA-TRAP: Yet another trivial RFID authentication protocol, in: 4th IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2006 Workshops), IEEE, 2006, pp. 640–643.

[24] M. Feldhofer, C. Rechberger, A Case Against Currently Used Hash Functions in RFID Protocols, in: On the Move to Meaningful Internet Systems, OTM'06, 2006.

[25] A. Bogdanov, G. Leander, C. Paar, A. Poschmann, M. Robshaw, Y. Seurin, Hash Functions and RFID Tags : Mind The Gap, in: 10th International Workshop Cryptographic Hardware and Embedded Systems, CHES'08, 2008.

[26] S. Kardas, S. Celik, M. Yildiz, A. Levi, PUF-enhanced offline RFID security and privacy, J. Network and Computer Applications 35 (6) (2012) 2059–2067.

[27] R. Pappu, Physical One-Way Functions, Ph.D. thesis, MIT (2001).

[28] S. Lim, I. Yie, Probabilistic privacy leakage from challenge-response RFID authentication protocols, in: Proceedings of the 7th Conference on 7th WSEAS International Conference on Applied Informatics and Communications (AIC'07), Vol. 7, Stevens Point, Wisconsin, USA, 2007, pp. 285–288.

[29] R. D. Pietro, R. Molva, An optimal probabilistic solution for information confinement, privacy, and security in RFID systems, J. Network and Computer Applications 34 (3) (2011) 853–863.

[30] L. Bolotnyy, G. Robins, Physically unclonable function-based security and privacy in RFID systems, in: International Conference on Pervasive Computing and Communications (PerCom 2007), IEEE Press, 2007, pp. 211–220.

[31] G. P. Hancke, Design of a secure distance-bounding channel for RFID, J. Network and Computer Applications 34 (3) (2011) 877–887.

[32] A. Mitrokotsa, C. Onete, S. Vaudenay, Mafia Fraud Attack against the RČ Distance-Bounding Protocol, in: IEEE International Conference on RFID-Technology and Applications (RFID TA 2012), IEEE Press, IEEE, Nice, France, 2012.

[33] A. Juels, S. Weis, Authenticating pervasive devices with human protocols, in: 25th Annual International Cryptology Conference, CRYPTO'05, 2005.

[34] N. Hopper, M. Blum, Secure human identification protocols, in: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, ASIACRYPT '01, Springer, 2001, pp. 52–66.

[35] H. Gilbert, M. Robshaw, H. Sibert, Active attack against HB+: a provably secure lightweight authentication protocol, Electronics Letters 41 (2005) 1169–1170.

[36] J. Munilla, A. Peinado, HB-MP: A further step in the HB-family of lightweight authentication protocols, Computer Networks 51 (2007) 2262–2267.

[37] H. Gilbert, M. Robshaw, Y. Seurin, HB#: Increasing the Security and Efficiency of HB+, in: Proceedings of the 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, EUROCRYPT'08, Springer-Verlag, 2008, pp. 361–378.

[38] J. Bringer, H. Chabanne, Trusted-HB: A Low-Cost Version of HB+ Secure Against Man-in-the-Middle Attacks, IEEE Transactions on Information Theory 54 (2008) 4339–4342.

[39] M. Safkhani, N. Bagheri, M. Naderi, S. Sanadhya, Security analysis of LMAP++, an RFID authentication protocol, International Conference for Internet Technology and Secured Transactions–ICITST'11 2011 (2011) 689–694.

[40] X. Leng, K. Mayes, K. Markantonakis, HB-MP+ protocol: An improvement on the HB-MP protocol, in: IEEE International Conference on RFID, IEEE, 2008, pp. 118–124.

[41] G. Hammouri, B. Sunar, PUF-HB: A Tamper-Resilient HB Based Authentication Protocol, in: ACNS, 2008.

[42] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, LMAP: A Real Lightweight Mutual Authentication Protocol for Low-cost RFID tags, in: Workshop on RFID Security, RFIDSec'06, 2006.

[43] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, M2AP: A Minimalist Mutual-Authentication Protocol for Low-cost RFID Tags, in: International Conference on Ubiquitous Intelligence and Computing, UIC'06, 2006.

[44] P. Peris-Lopez, J. C. Hernandez-Castro, J. M. Estevez-Tapiador, A. Ribagorda, EMAP: An Efficient Mutual Authentication Protocol for Low-Cost RFID Tags, in: OTM Federated Conferences and Workshop: IS Workshop, IS'06, 2006.

[45] T. Li, G. Wang, Security Analysis of Two Ultra-Lightweight RFID Authentication Protocols, in: 22nd International Information Security Conference, IFIP SEC'07, 2007.

[46] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, Breaking LMAP, in: Conference on RFID Security, 2007.

[47] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, Passive Attack Against the M2AP Mutual Authentication Protocol for RFID Tags, in: First International EURASIP Workshop on RFID Technology, 2007.

[48] M. Bárász, B. Boros, P. Ligeti, K. Lója, D. Nagy, Breaking EMAP, in: Conference on Security and Privacy for Communication Networks – SecureComm 2007, 2007.

[49] H.-Y. Chien, SASI: A New Ultralightweight RFID Authentication Protocol Providing Strong Authentication and Strong Integrity, IEEE Transactions on Dependable and Secure Computing 4 (2007) 337–340.

[50] T. Van Le, M. Burmester, B. De Medeiros, Universally composable and forward-secure RFID authentication and authenticated key exchange, in: Proceedings of the 2nd ACM symposium on Information, computer and communications security, ACM, 2007, pp. 242–252.

[51] M. Burmester, J. Munilla, Lightweight RFID authentication with forward and backward security, ACM Transactions on Information and System Security (TISSEC) 2011 14.

[52] Y. Hanatani, M. Ohkubo, S. Matsuo, K. Sakiyama, K. Ohta, A Study on Computational Formal Verification for Practical Cryptographic Protocol: The Case of Synchronous RFID Authentication, in: Financial Cryptography Workshops, 2011, pp. 70–87.

[53] M. Brusò, K. Chatzikokolakis, J. den Hartog, Formal Verification of Privacy for RFID Systems, in: CSF, IEEE, 2010, pp. 75–88.

[54] H. S. Kim, J.-H. Oh, J.-B. Kim, Y.-O. Jeong, J.-Y. Choi, Formal Verification of Cryptographic Protocol for Secure RFID System, in: NCM (2), IEEE, 2008, pp. 470–477.

[55] M. Asadpour, M. T. Dashti, A Privacy-friendly RFID Protocol using Reusable Anonymous Tickets, in: 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom'11, IEEE, 2011, pp. 206–213.

[56] J. Garcia-Alfaro, J. Herrera-Joancomartí, J. Melià-Seguí, Practical Eavesdropping of Control Data from EPC Gen2 Queries with a Programmable RFID Toolkit, Hakin9[Online].

[57] B. Barak, S. Halevi, A model and architecture for pseudo-random generation with applications to/dev/random, in: Proceedings of the 12th ACM conference on Computer and communications security, CCS '05, ACM, 2005, pp. 203–212.

[58] M. Bellare, B. Yee, Forward-security in private-key cryptography, Topics in Cryptology–CT-RSA'03 2612 (2003) 1–18.

[59] A. Desai, A. Hevia, Y. Yin, A practice-oriented treatment of pseudorandom number generators, in: Advances in Cryptology, EUROCRYPT'02, Springer, 2002, pp. 368–383.

[60] M. Feldhofer, J. Wolkerstorfer, V. Rijmen, AES Implementation on a Grain of Sand, IEE Proceedings – Information Security 152 (1) (2005) 13–20.

[61] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, S. Chee, HIGHT: A New Block Cipher Suitable for Low-Resource Device, in: L. Goubin, M. Matsui (Eds.), Cryptographic Hardware and Embedded Systems – CHES 2006, Vol. 4249 of Lecture Notes in Computer Science, Springer, Yokohama, Japan, 2006, pp. 46–59.

[62] C. De Canniere, B. Preneel, Trivium Specifications, Tech. rep., Ecrypt 2008, [Online, last access Apr. 2012] Available at `http://www.ecrypt.eu.org/stream/triviumpf.html`.

[63] M. Hell, T. Johansson, W. Meier, Grain: a stream cipher for constrained environments, International Journal of Wireless and Mobile Computing, 2 (1) (2007) 86–93.

[64] P. Peris-Lopez, J. Hernandez-Castro, J. Estevez-Tapiador, A. Ribagorda, LAMED a PRNG for EPC Class-1 Generation-2 RFID specification, Computer Standards & Interfaces (2007) 88–97.

[65] J. Melia-Segui, J. Garcia-Alfaro, J. Herrera-Joancomarti, Multiple-polynomial LFSR based pseudorandom number generator for EPC Gen2 RFID tags, in: 37th Annual Conference on IEEE Industrial Electronics Society (IECON 2011), IEEE, 2011, pp. 3820–3825.

[66] A. Menezes, P. Van Oorschot, S. Vanstone, Handbook of applied cryptography, CRC Press Inc., 1997.

[67] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation,

in: Proceedings of 38th Annual Symposium on Foundations of Computer Science, FOCS'97, 1997, pp. 394–403.
[68] M. Abdalla, M. Bellare, Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques, Advances in Cryptology–ASIACRYPT'00 1976 (2000) 546–559.
[69] R. Küsters, T. Truderung, Reducing Protocol Analysis with XOR to the XOR-Free Case in the Horn Theory Based Approach, J. Autom. Reasoning 46 (3-4) (2011) 325–352.
[70] S. Escobar, C. Meadows, J. Meseguer, Maude-NPA: Cryptographic Protocol Analysis Modulo Equational Properties, in: FOSAD, 2007, pp. 1–50.
[71] D. Basin, S. Mödersheim, L. Vigano, OFMC: A symbolic model checker for security protocols, International Journal of Information Security 4 (2005) 181–208.
[72] C. Arora, M. Turuani, Validating Integrity for the Ephemerizer's Protocol with CL-Atse, in: Formal to Practical Security, 2009.
[73] C. J. F. Cremers, P. Lafourcade, P. Nadeau, Comparing State Spaces in Automatic Security Protocol Analysis, in: Formal to Practical Security, 2009, pp. 70–94.
[74] S. Kremer, Modelling and analyzing security protocols in cryptographic process calculi, Ph.D. thesis, École normale supérieure de Cachan-ENS Cachan (2011).
[75] A. Armando, W. Arsac, T. Avanesov, M. Barletta, A. Calvi, A. Cappai, R. Carbone, Y. Chevalier, L. Compagna, J. Cuéllar, et al., The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures, in: Proceedings of the 18th international conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'12, Springer, 2012, pp. 267–282.
[76] Y. Chevalier, L. Vigneron, Rule-based Programs Describing Internet Security Protocols, Electronic Notes in Theoretical Computer Science 124 (2005) 113–132.
[77] A. Armando, L. Compagna, Satmc: A sat-based model checker for security protocols, Logics in Artificial Intelligence (2004) 730–733.
[78] O. K. Y. Boichut, P.-C. Heam, Automatic Verification of Security Protocols Using Approximations, Tech. rep., INRIA Research Report (2005).
[79] B. Song, C. J. Mitchell, RFID authentication protocol for low-cost tags, in: Proceedings of the first ACM conference on Wireless network security, WiSec'08, 2008.
[80] C. H. Lim, T. Kwon, Strong and robust RFID authentication enabling perfect ownership transfer, in: Conference on Information and Communications Security, ICICS'06, 2006.
[81] M. O. Koutarou, K. Suzuki, S. Kinoshita, Cryptographic Approach to "Privacy-Friendly" Tags, in: In RFID Privacy Workshop, 2003.
[82] M. Bellare, T. Krovetz, P. Rogaway, Luby-Rackoff backwards: Increasing security by making block ciphers non-invertible, Advances in Cryptology–EUROCRYPT'98 1403 (1998) 266–280.

Appendix A. Security of block ciphers

A block cipher is a function $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ which transforms an n-bit message block x into an n-bit string y under the control of k-bit key $k : y = E(k, x)$. The function is invertible in the sense that for each key the map $E_k \stackrel{def}{=} E(k, .)$ is a permutation of $\{0,1\}^n$, and the knowledge of k permits the computation of ${E_k}^{-1}$. In typical usage, a random key k is chosen and kept secret between a pair of users. The function E_k is then used by the two parties to process data in some way before they send it to each other.

The security provided by a block cipher depends on the security against key recovery and the security of the pseudorandom permutation E. It should be hard to distinguish the input/output behaviour of E_K from a random function without knowing the key K. We fix a block cipher $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ with key-size k and block size n.

Appendix A.1. Security against key recovery

Classically, the security of block ciphers has been related to key recovery. That is, the analysis of a block cipher E is done by considering some number q of inputs and outputs examples $(M_1, C_1), \ldots, (M_q, C_q)$, and trying to find K. K is a random, unknown master key and $C_i = E_K(M_i)$ for $i = 1, \ldots, q$ and $M_1, \ldots, M_q$ are all distinct n-bit strings. The question is how hard is it for an attacker to find a master key K? Some typical attack strategies are considered in this scheme are named Known-Plaintext Attack KPA and Chosen-Plaintext Attack CPA. In the first attack, $M_1, \ldots, M_q$ are distinct, arbitrary and are not controlled by the adversary algorithm. However by analysing a given block of plaintext and corresponding ciphertext, the attacker tries to extract useful information for the recovery of plaintext encrypted in different ciphertexts or secret keys. In the second, $M_1, \ldots, M_q$ are adaptively picked by the adversary algorithm. Given its ability to choose plaintexts and generating corresponding ciphertexts, the adversary algorithm accesses an *oracle* of the function E_K and feeds the oracle M_1, then gets back $C1 = E_K(M_1)$. The latter value lets the adversary adaptively decide on the value M_2. Thus, it feeds the *oracle* to get back $C2$, and so on. Even if CPA gives the adversary more power, these latter are not always realistic in practise. The generic and most used attack strategy that works against any block cipher is named exhaustive key search EKS or a brute-force attack. This attack always returns the corresponding key with the above sample of input-output. In the worst case, an adversary uses 2^k computations of the block cipher to obtain the key. We can conclude that there is no block cipher which is perfectly secure. It is viewed as secure if the best key recovery attack is computationally infeasible, which means that it requires a value of queries q or a running time too large to make the attack practical. Thus, to make key recovery by EKS computationally infeasible, one must enlarge the key length k of the block cipher.

Exhaustive key search attack (EKS)

The adversary algorithm tries all possible keys $K' \in \{0,1\}^k$ until it finds the one that explains the input-output pairs. Let $K_1, \ldots, K_2^k$ be a list of all K-bits keys. Let $K \stackrel{\$}{\rightarrow} \{0,1\}^k$ be the searched key and let (M_1, C_1) an example that satisfies $E_K(M_1) = C_1$.

Exhaustive Key search based on one sample
algorithm $EKS_E(M_1, C_1)$
for $i = 1, \ldots 2^k$ ***do***
if $E(K_i, M_1) = C_1$ ***then return*** K_i

The likelihood of the attack returning the searched key can be increased by testing more samples of input-output. Moreover, the computations can be performed in parallel.

Security against key-recovery is necessary but it has been proved that it is not sufficient for block ciphers as it can still be possible for an attacker to find a relation between the input and the output after some time of executions without knowing K.

Appendix A.2. Security of the pseudorandom permutation

The security of a block cipher is defined by the advantage an adversary has in distinguishing a real function E_k from a random permutation (cf. Appendix C). Using even a very *good* block cipher, the encryption (e.g., under the common mode of operation CBC (Cipher Block Chaining)) becomes insecure once $2^{n/2}$ blocks are encrypted under the same master key [82]. At this point, partial information about the message begins to leak. This leads to what is named birthday attacks [67]. For example direct use of 64-bit block cipher usually enables to safely encrypt no more than 2^{32} blocks. The cost of such an attack depends only on the block length. As a solution, we can enlarge the block length n, so that the $2^{n/2}$ time is unpractical. However, this is not a practical solution for environment supporting devices with limited memory capacities.

Appendix B. Relevant improvement schemes

To overcome the limitation of birthday attacks and securely encrypt more than $2^{n/2}$ messages in block ciphers, two major improvements are shown in the literature: the master-key re-keying and the data-dependent re-keying.

The master-key re-keying. Authors in [68] propose a scheme that protects against birthday attacks by changing the master key before the threshold number of encryptions permitting the attack is reached. The results show that re-keying the master key every $2^{n/3}$ encryptions, increases the threshold to $2^{2n/3}$ encryptions. This means that with the re-keying scheme, one can safely encrypt more data. This re-keying scheme also minimises the amount of damage that might be caused by key exposure. The exposure of the current key could determine all future keys (if the adversary has followed all the transactions), but if well used, the system cannot reveal past master keys that have to remain computationally infeasible to predict for the adversary even given the current master key and state.

The data-dependent re-keying. To minimise the advantage of an adversary to recognise encrypted data with some generated keys particularly when $q \geqslant 2^{n/2}$, [82] show that E must not be a family of permutation. The idea is to turn a pseudo random permutation (e.g., a block cipher) to a pseudorandom function which has not to be invertible. The basic case of the proposition apply when the key k and the input x of the block cipher have the same length $k = n$. The changed function F is then defined as $F(k, x) = E(E(k, x), x)$, where E is the permutation function. Thus, $F_k(x) = E_{k'}$, where $k' = E_k(x)$. This change is called *data-dependent re-keying*. F is twice the cost of computing E since there are two applications of E for each application of F. Protocols that are not worried about computing cost can use this scheme. A general construction of the solution is found in [82].

Appendix C. Quality of block ciphers

The quality of a given block cipher is captured by a function $Sec_E(q, t)$ which returns the maximum *advantage* that an adversary A can obtain in distinguishing a real function *Real* E_k from a random function *Rand* if A has seen q input/output examples and is allowed computational resources bounded by t (i.e., in the complexity-theoretic model, t will bound computing time). For more details see [82]. The advantage is a number between 0 and 1 given as the difference of two probabilities: the probability that the adversary outputs 1 given a *Real* random function E_k from a family of block-ciphers E, and the probability that the adversary outputs 1 given a random permutation *Rand* (cf. reference [82]).

We denote by: $Pr[Rand^A_{Range(E)} \Rightarrow r]$, the probability that A outputs r. Then,

$Adv_E(A) = Pr[Real^A_E \Rightarrow 1] - Pr[Rand^A_{Range(E)} \Rightarrow 1]$ and

$Sec_E(q, t) = max_A Adv_E(A)$.

The second term is the maximum advantage over all adversaries A having time-complexity at most t and making at most q oracle queries. The time complexity is per convention, the total worst-case execution time of the experiment underlying the first term $Adv_E(A)$, plus the size of the code of A.

Thus the adversary advantage depends on its strategy and the resources he has, namely the running time t and the number q of oracle queries. The result of the first term is a number between -1 and 1. An advantage which is close to 1 means that A is a good algorithm or/and E is not secure whereas an advantage that is close to 0 means that A is a poor algorithm or/and E resists the attacks mounted by A. Therefore, E is a secure pseudorandom permutation function if $Adv_E(A)$ is *small* for all A that uses reasonable amounts of resources.